ARMY RESEARCH LABORATORY

Web-Based Programming for Real-Time News Acquisition

by Andrew M. Neiderer and John Richardson

ARL-MR-671 September 2007

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5067

ARL-MR-671 **September 2007**

Web-Based Programming for Real-Time News Acquisition

Andrew M. Neiderer and John Richardson
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			
1. REPORT DATE *(DD-MM-YYYY)* September 2007	**2. REPORT TYPE** Final		**3. DATES COVERED (From - To)** October 2006–February 2007
4. TITLE AND SUBTITLE Web-Based Programming for Real-Time News Acquisition			**5a. CONTRACT NUMBER**
			5b. GRANT NUMBER
			5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Andrew M. Neiderer and John Richardson			**5d. PROJECT NUMBER** 611102H48
			5e. TASK NUMBER
			5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: AMSRD-ARL-CI-CT Aberdeen Proving Ground, MD 21005-5067			**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-MR-671
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			**10. SPONSOR/MONITOR'S ACRONYM(S)**
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			
13. SUPPLEMENTARY NOTES			
14. ABSTRACT This report describes a Web 2.0 application that was developed at the U.S. Army Research Laboratory in support of its Real-Time News Analysis (RTNA) project. It uses the Google, Inc. AJAX search application programming interface to acquire data and subsequently formats resultant data for analysis. News stories for a specified topic (e.g., terrorist bombing) are gathered from public sources by a function in a JavaScript node of an extensible markup language formatted document (XHTML). Content of selected elements is then extracted, or "scraped," from the XHTML. The designed graphical user interface allows one to choose up to 10 words and/or phrases and permits explicit exclusion of certain semantics. Presently, the selected data sources are determined by Google News and user-specified in a Google Web service. A Google gadget for Maps has been added for geographic visualization of location, and additional searchers for Google Video, Blog, and Book have been tested and can be easily added to the search controller. The application also allows for integration of asynchronous JavaScript and XML technology, including Java servlets for requesting data and Java Server Pages for the responses.			
15. SUBJECT TERMS extensible markup language (XML), Google AJAX, Google Maps, Google News, Google Web, hypertext markup language (HTML), JavaScript, Social Network Analysis, mash-up			

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Andrew M. Neiderer
a. REPORT UNCLASSIFIED	**b. ABSTRACT** UNCLASSIFIED	**c. THIS PAGE** UNCLASSIFIED	UL	36	**19b. TELEPHONE NUMBER** (*Include area code*) 410-278-3203

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures iv

1. Introduction 1

2. RTNA Graphical User Interface 3

3. Google AJAX Searching for RTNA 4

4. Content Extraction 6

5. Conclusion and Future Considerations 6

6. References 8

Appendix A. The RTNA XHTML 1.0 Document 9

Appendix B. Content Extraction Software 23

List of Symbols, Abbreviations, and Acronyms 27

Distribution List 28

List of Figures

Figure 1. Real-time news analysis (RTNA) GUI example with a Google Maps gadget. The satellite view of the area includes a marker, which is a location within Fallujah, Iraq. 2

Figure 2. Enabled state for 1, 4, and 8 (or more) search results. 3

Figure 3. Disabled state for 1, 4, and 8 (or more) search results. 3

Figure 4. Icon defined for sorting of Google News by date, or selection of center point for the Google Maps service...4

Figure 5. Sample results for a “Al-Sadr” OR “Mahdi Army” news search at a given time. 5

1. Introduction

An interpretation of a news event from a single source can result in an erroneous conclusion, especially if little (or nothing) is known about the validity of that source. For example, one person's freedom-fighter can be another's terrorist (*1*). Thus, it is often better if one has access to a broad collection of data, and is able to select specific reporting media. Software for the acquisition of real-time news published on the Internet has been developed at the U.S. Army Research Laboratory (ARL).

The maturation of the semantic Web in combination with a Google, Inc. AJAX search using its application programming interface (API) are two key developments that are exploited in ARL's rapid collection of current world event data. One reason for developing this tool is that news (both local and distant) on a particular topic and subsequent responses can propagate quickly. Therefore, an attempt is made to provide the analyst with an aggregation of information on a subject as it becomes publicly available.

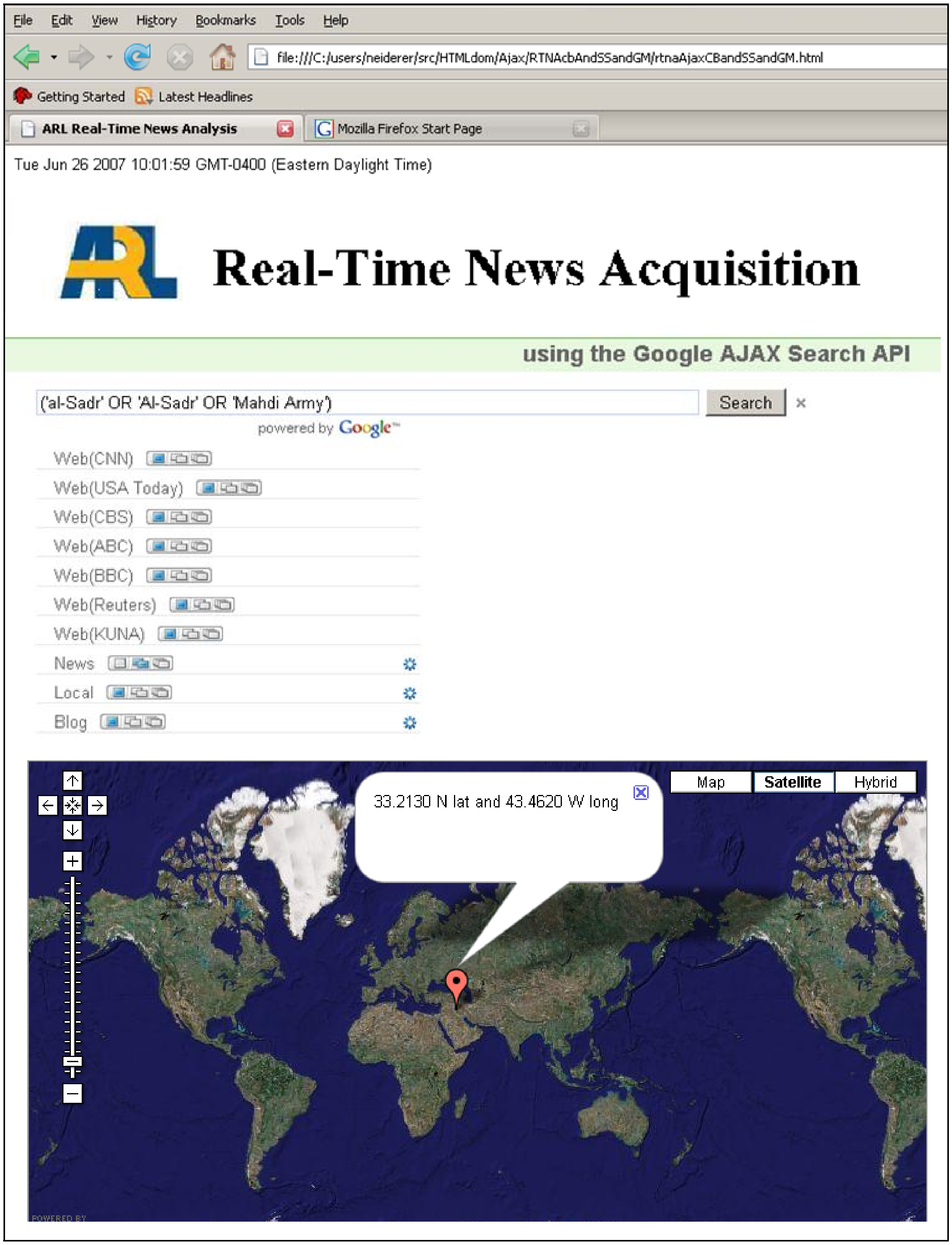
Development involved consideration of all six Google services as potential data sources: Google News, Web, Maps, Video, Book, and Blog. The first three (News, Web, and Maps) are being used for the collection of data. The Google News site (http://news.google.com) uses over 10,584 sources (see http://blog.outer-court.com/googlenews/), including both national and international nodes. A Google Web search requires one to designate a particular uniform resource locator (URL), while an even more specific Maps search involves the actual specification of latitude and longitude for the point of interest.

Google Blog, Video, and Book are not included in the search controller at this time. Google Book is not pertinent to real-time analysis. In a Google Video search, "hits" result in images or illustrations, which are inconsistent with our intention in gathering news and not interpreting it. Likewise, blogs (i.e., Web logs) tend to be opinionated and are not professionally reported (*2*).

The following sections provide technical details of information acquisition. The graphical user interface (GUI), which is illustrated in figure 1, is first presented and described. The next section gives an overview of asynchronous JavaScript and XML (Ajax),* where Google's API is used to find the most recent news. Section 4 examines the method selected for scraping of data. The conclusion offers future considerations for enhanced, capabilities.

The attached appendices include the actual XHTML, JavaScript, and Java code for the complete Web application. The cascading style sheet used in the XHTML can be obtained from Google (http://www.google.com/uds/css/gsearch.css).

*Google, Inc. uses the term AJAX throughout the description of its search API. The authors of this paper prefer Ajax, which was coined by the originator Jesse James Garret. The reason is that our approach encompasses technologies that is consistent with Ajax design.



File Edit View History Bookmarks Tools Help
file:///C:/users/neiderer/src/HTMLdom/Ajax/RTNAcbAndSSandGM/rtnaAjaxCBandSSandGM.html
Getting Started
Latest Headlines
ARL Real-Time News Analysis
Mozilla Firefox Start Page
Map
Satellite
Hybrid
POWERED BY

Tue Jun 26 2007 10:01:59 GMT-0400 (Eastern Daylight Time)

Real-Time News Acquisition

using the Google AJAX Search API

('al-Sadr' OR 'Al-Sadr' OR 'Mahdi Army') Search ×

powered by Google

Web(CNN)
Web(USA Today)
Web(CBS)
Web(ABC)
Web(BBC)
Web(Reuters)
Web(KUNA)
News
Local
Blog

33.2130 N lat and 43.4620 W long

Figure 1. Real-time news analysis (RTNA) GUI example with a Google Maps gadget. The satellite view of the area includes a marker, which is a location within Fallujah, Iraq.

2. RTNA Graphical User Interface

The results of a sample search using the Mozilla Firefox 1.5 browser for display are illustrated in figure 1. At the very top of the GUI is the date and time the query was made. This is computed in a static JavaScript node of the XHTML document; static JavaScript is run automatically when the Web browser is loading. Each instance of a browser exposes many JavaScript objects, including date, in determination of the document object model (DOM), which is an abstract interface that allows for manipulation of the browser.

The date object uses getMonth(), getDate(), and getFullYear() methods when constructing the string in the JavaScript node. Note also that the DOM includes events and its handlers (see Flanagan (*3*) for a thorough discussion of the JavaScript interface to the DOM of browsers).

The GUI defines a search box that handles up to 10 tokens, where a token is defined as a string of characters surrounded by white space. The format for an entry, including stop words, is described in the book by Calishain and Dornfest (*4*). When the user is satisfied with a particular query, a search is started by pressing "Search." This button is a text field of the XHTML <form>, i.e., a child node <input> of type "button" and value "Search." By using a button the DOM of the browser avoids a complete page reload, resulting in much better response.

The results of the request are then available. A result bar (see figures 2 and 3) exists for each Google service (remember that only Web, News, and Maps searchers have been added to the controller). In this example, a total of five Google Web searches and a Google News search are displayed. A Google Web search object (GwebSearch()) is necessary for each site selected; here, URLs for ABC, CBS, NBC, CNN, and Reuters news were chosen. Google News uses an algorithm for a much more vast selection of sites throughout the world.

Figure 2. Enabled state for 1, 4, and 8 (or more) search results.

Figure 3. Disabled state for 1, 4, and 8 (or more) search results.

There is also the option for a chronological sort of Google News or setting the center point for a Google Maps, or Local, search. This involves invocation of the appropriate functions for GnewsSearch (setResultOrder() method) and GlocalSearch (setCenter() method) objects, respectively, by simply clicking the icon (figure 4) and confirming the selection.

Figure 4. Icon defined for sorting of Google News by date, or selection of center point for the Google Maps service.

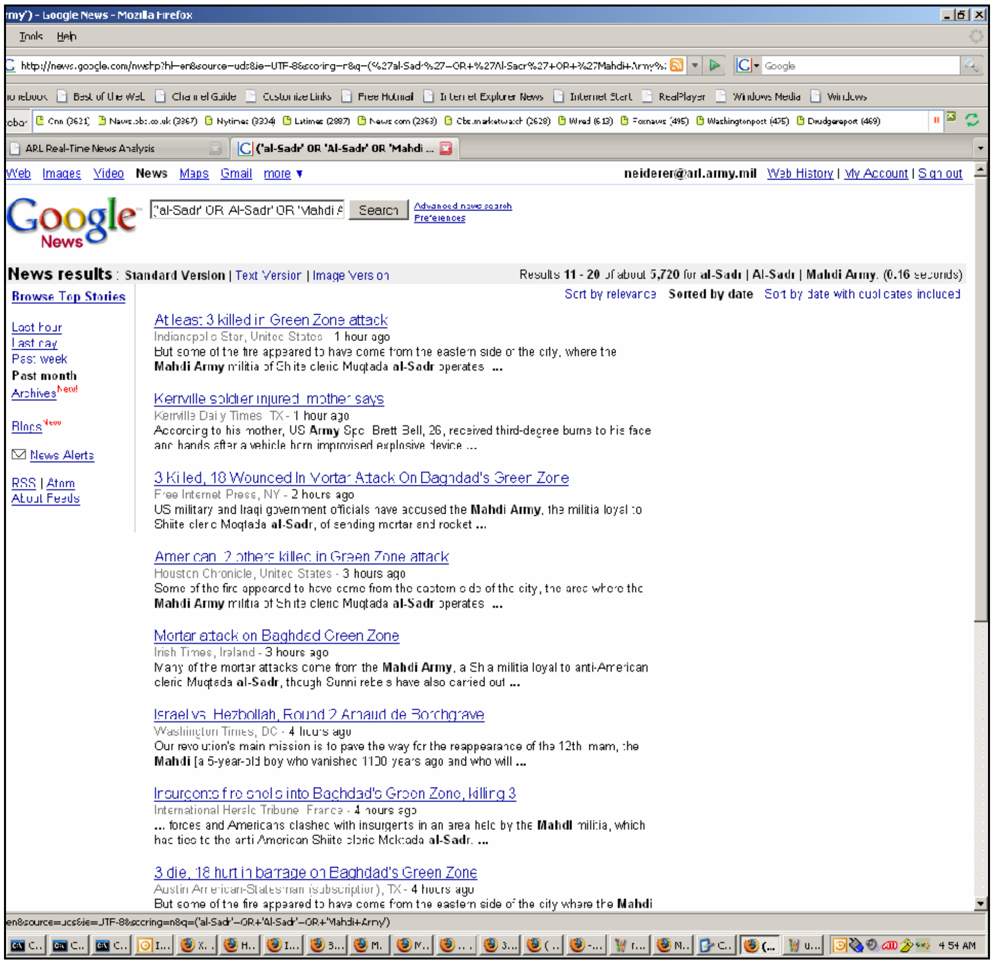
Finally, a Google gadget for a Maps mash-up has been added to the XHTML. The intent is a two-dimensional (2-D) visual display of an area by providing the latitude and longitude of the map center point. An accuracy of 10^{-6} decimal degrees (e.g., designation of a specific building) is possible if Google data exists for that point. A search of the Google Local database is possible using its Maps API (see http://local.google.com), but here we only provide a display. An excellent discussion of both Google Maps and Google Earth is available in the book by Brown (*5*).

The map panel in figure 1 shows the current view (an orthographic projection when viewing from infinity on the positive z axis). At the top-left corner of this map panel is zoom and navigation controls. Instead of using defined increments, the user may chose to click and drag the map in any direction. At the other upper corner are view controls to switch between map and satellite; hybrid is a combination of both map and satellite.

Partial results for a figure 1 query are shown in figure 5. Note that a chronological sort had been specified, and that keywords are in bold-face type within the snippet. Also recall that an entire article is retrieved by simply clicking on the hypertext.

3.   Google AJAX Searching for RTNA

A key for an AJAX search using the Google API can be obtained at http://code.google.com/apis /ajaxsearch/signup.html. The key value is necessary for a Google AJAX search when the <script> node in the XHTML document is defined; this same attribute value is valid for Google Maps access (see appendix B, which provides the XHTML for RTNA). The result is a dynamic, highly-responsive application that runs on your desktop but with all the advantages of being connected to the Internet.



News results : Standard Version | Text Version | Image Version

Results 11 - 20 of about 5,720 for al-Sadr | Al-Sadr | Mahdi Army. (0.16 seconds)

Sort by relevance   Sorted by date   Sort by date with duplicates included

[At least 3 killed in Green Zone attack](#)
Indianapolis Star, United States - 1 hour ago
But some of the fire appeared to have come from the eastern side of the city, where the **Mahdi Army** militia of Shiite cleric Muqtada **al-Sadr** operates ...

[Kerrville soldier injured, mother says](#)
Kerrville Daily Times, TX - 1 hour ago
According to his mother, US **Army** Spc. Brett Bell, 26, received third-degree burns to his face and hands after a vehicle-born improvised explosive device ...

[3 Killed, 18 Wounded In Mortar Attack On Baghdad's Green Zone](#)
Free Internet Press, NY - 2 hours ago
US military and Iraqi government officials have accused the **Mahdi Army**, the militia loyal to Shiite cleric Moqtada **al-Sadr**, of sending mortar and rocket ...

[American, 2 others killed in Green Zone attack](#)
Houston Chronicle, United States - 3 hours ago
Some of the fire appeared to have come from the eastern side of the city, the area where the **Mahdi Army** militia of Shiite cleric Muqtada **al-Sadr** operates ...

[Mortar attack on Baghdad Green Zone](#)
Irish Times, Ireland - 3 hours ago
Many of the mortar attacks come from the **Mahdi Army**, a Shia militia loyal to anti-American cleric Muqtada **al-Sadr**, though Sunni rebels have also carried out ...

[Israel vs. Hezbollah, Round 2 Arnaud de Borchgrave](#)
Washington Times, DC - 4 hours ago
Our revolution's main mission is to pave the way for the reappearance of the 12th Imam, the **Mahdi** [a 5-year-old boy who vanished 1100 years ago and who will ...

[Insurgents fire shells into Baghdad's Green Zone, killing 3](#)
International Herald Tribune, France - 4 hours ago
... forces and Americans clashed with insurgents in an area held by the **Mahdi** militia, which has ties to the anti-American Shiite cleric Moktada **al-Sadr**. ...

[3 die, 18 hurt in barrage on Baghdad's Green Zone](#)
Austin American-Statesman (subscription), TX - 4 hours ago
But some of the fire appeared to have come from the eastern side of the city where the **Mahdi**

Figure 5. Sample results for a “Al-Sadr” OR “Mahdi Army” news search at a given time.

There are many situations where one benefits from Ajax effects: e.g., DOM access of browser using the JavaScript interface to the XHTML. But the actual asynchronous communication with a server is accomplished by:

1. creation of the request object for communicating with a server,
2. telling the Web browser which JavaScript function to run when the request object ready state is 4 (see reference 3 for a complete description of ready states and status codes),
3. connecting to Web server for communication, and
4. the actual request to connect to Web server.

If these steps are satisfied, then the JavaScript request object can communicate with the Web server asynchronously. All five papers of an Ajax tutorial by McLaughlin (*6*) can be found at this URL.

Note that the previous four steps are taken care for us by the Google AJAX search API. The intent here is to make the user aware of what is happening behind the scenes in a search, and to assist in future additions using an Ajax design. The XHTML in appendix B includes static JavaScript for determining the request object of a browser: a Microsoft ActiveXObject object for Internet Explorer or an XMLHttpRequest object for a Mozilla-based browser.

4. Content Extraction

The intention of RTNA is to provide the text content of a news report as data. A news report discovered using the search functionality of RTNA is typically embedded in a document written in HTML. As a result, a mechanism is needed to find and extract the news report from this document. The content extraction software developed for RTNA uses a HTML DOM parser to locate and extract text content from HTML documents.

An HTML DOM parser transforms an HTML document into a tree structure with nodes representing tags (element node) and text contained in the tags (text node). The content-extraction software developed for RTNA uses the CyberNeko HTML Parser (http://people.apache.org/~andyc/neko/doc/html/) to create the document tree. The content-extraction software recursively traverses the document tree to gather the text from text nodes. The gathered text is returned as the content of the document. The text gathering is governed by one heuristic: an element node that is unlikely to contain any part of the news report is removed from the document tree. To illustrate, the element node representing a <script> tag is always removed from the document tree. The text nodes of this element are likely to contain unwanted JavaScript content, instead of news report content. Additional heuristics to govern content extraction are the subject of future research.

5. Conclusion and Future Considerations

Currently, ARL's RTNA gathers news for a user-specified topic using three of the six Google services available. A Google News search typically results in the very latest from some 10,584 sources around the world. On the other hand a Google Web search requires actual specification of the URL for the news source, which also means more control of a search by selecting a particular source; this is accomplished by using the setSiteRestriction() method of a GwebSearch() object for a URL.

A Google search within the XHTML document makes use of a request/response model typical of Ajax technology. Searches are done asynchronously from the client browser. For example, the response may be a Java Server Page, which is typically an HTML document, generated by a servlet running on the server. The result is a Web 2.0 application with a very large set of resources but runs like a desktop application.

Now that this initial version is stable, additional capabilities are being considered. One should be fully aware of the objectivity in the story. For example, a story from a primary wire service is typically repeated to the local level; perhaps a social network analysis (SNA) of an individual story would be instructive in seeing if/how the story has evolved (use of the JavaScript Date object within the XHTML will assist in this). Also note that SNA has been done for a particular news media (*7*) but the approach will be considered for inclusion. The other Google services (Blogs, Video, and Book) will also be further examined for inclusion as well. Lastly, we plan on investigating a scaleable vector graphics, which is just 2-D XML, display of public opinion on a particular topic with time from a <script> in our XHTML; for example, something similar to, or including, the spatial analysis of news as done at the State University of New York (*8*).

6. References

1. Corman, S. R.; Dooley, K. J. *Fighting Terrorist*; Arizona State University: Tempe, AZ, 85287, 2001–2002.
2. Lloyd, L.; Kaulgud, P.; Skiena, S. *Newspapers vs. Blogs: Who Gets the Scoop*? Department of Computer Science, State University of New York at Stony Brook: Stony Brook, NY, 11794–4400.
3. Flanagan, D. *JavaScript: The Definitive Guide*; 4th ed.; O'Reilly & Associates, Inc.: Sebastopol, CA, 2002.
4. Calishain, T.; Dornfest, R. *Google Hacks*; 2nd ed.; O'Reilly & Associates, Inc.: Sebastopol, CA, 2005.
5. Brown, M. C. *Hacking Google Maps and Google Earth*; Wiley Publishing, Inc.: Indianapolis, IN, 2006.
6. McLaughlin, B. Mastering Ajax, Part 3: Advanced Requests And Responses in Ajax. http://www-128.ibm.com/developerworks/web/library/wa-ajaxintro3/, accessed February 2006.
7. Batagelj, V.; Mrvar, A. *Density Based Approaches to Network Analysis*. Analysis of Reuters Terror News Network. University of Ljubljana, 2001.
8. Mehler, A.; Bao, Y.; Li, X.; Wang, Y.; Skiena, S. Spatial Analysis of News Sources. *IEEE Transactions on Visualization and Computer Graphics*, State University of New York at Stony Brook: Stony Brook, NY, **2006**, 12 (*5*).

Appendix A. The RTNA XHTML 1.0 Document

The extensible hypertext markup language (XHTML), including the internal cascading style sheet (CSS) and JavaScript, for ARL's RTNA are now included. The external CSS can be found at the Google site http://www.google.com/uds/css/gsearch.css. Also both a Google AJAX search API and Maps key must be defined; both are freely available after registering at ttp://code. google.com/apis/ajaxsearch/signup.html and http://www.google.com/maps/api_signup, respectively.

This appendix appears in its original form, without editorial change.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html lang="en-CA" xml:lang="en-CA" xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <!-- ARL Real-Time News Analysis.                                        -->
    <!--                                                                     -->
    <!--  by Andrew M. Neiderer, 21 February 2007.                           -->
    <!--                                                                     -->
    <!--  Note -                                                             -->
    <!--   (1) callback code was borrowed and modified from Google Web site  -->
    <!--  http://code.google.com/apis/ajaxsearch/documentation/#SearchControlCallbacks, -->
    <!--   (2) Google Map code originally by Doug Henderson                  -->
    <!--  http://www3.telus.net/DougHenderson/.                              -->

    <meta http-equiv="content-type" content="text/html; charset=ISO-8859-1"/>
    <meta http-equiv="window-target" content="_top"/>

    <title>
      ARL Real-Time News Analysis
    </title>

    <!-- external and internal CSS -->

    <link href="http://www.google.com/uds/css/gsearch.css"
          type="text/css" rel="stylesheet"/>

    <style type="text/css">
      body *, table *,
      body {
        font-family: Arial,Sans-serif;
        font-size: 13px;
      }

      title {
        text-align : center
      }

      img {
        align : middle
      }

      h1 {
        font-size : 18px;
        font-weight : bold;
        background-color : rgb(230,248,221);
        border-top : 1px solid rgb(128,198,90);
        text-align : center;
        margin-bottom : 10px;
        padding-bottom : 4px;
      color : #676767;
      text-align : center
    }
    h1 .tagline,
    h1 a .tagline {
      font-size : 13px;
      font-weight : normal;
      color : #676767;
      text-decoration : underline;
      cursor : pointer;
    }

    td {
      vertical-align : top;
    }
    td.searchControl {
      padding-left : 25px;
      width : 700px;
    }
    td.map {
      width: 550px;
    }

    #mapDiv {
      border : 1px solid #979797;
      width : 100%;
      height : 400px;
    }

    .gsc-keeper {
      display : none;
    }
    .gsc-localResult .gsc-keeper {
      display : block;
    }

    <!-- over-ride rule for gsearch.css; change width of -->
    <!--  search box: 2/7/07, Andrew M. Neiderer.        -->
    .gsc-control {
      width : 600px;
    }
    table.gsc-search-box {
      width : 600px;
    }
  </style>

  <!-- external and internal JavaScript which includes support of Ajax request/response  -->
  <!--  modeling.  Remember that static JavaScript, ie definitions outside a             -->
  <!--  function, is run automatically when the Web browser is loading.                  -->

  <!-- Google AJAX search API key -->
```

```
    <script src="http://www.google.com/uds/api?file=uds.js&amp;v=1.0&amp;key=ABQIAAAAZlKOslBCsTw_aBLk4taNOxQ5JOKSYN5SawKW
            type="text/javascript">
    </script>

    <!-- Google Maps API key -->
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=api&amp;v=2&amp;key=ABQIAAAAZlKOslBCsTw_aBLk4taNOxQ
            type="text/javascript">
    </script>
    <script src="rot13.js"
            type="text/javascript">
    </script>

    <script type="text/javascript">
     //<![CDATA[
      var date = new Date();
      document.write(date.toString());

      function googleDate(date)
      {
        var month = ['January','February','March','April','May','June','July',
                     'August','September','October','November','December'];

        return month[date.getMonth()] + ' ' +
         date.getDate() + ', ' + date.getFullYear();
      }

      var searchString = "('al-Sadr' OR 'Al-Sadr' OR 'Mahdi Army') AND February AND 16 AND 2007";
//      var searchString = "('al-Sadr' OR 'Al-Sadr' OR 'Mahdi Army') AND '" + googleDate(date) +"'";

                                         // Ajax
      var request = null;                //  request object
      var requestURL =                   //  request URL
       getUpdatedBoardSales-ajax.php;
      var response = null;               //  response from server
      var lastXMLresponse = null;

      var success = false;

      // create request object statically for talking to Web server

      try {
        // Mozilla-based browsers
        request = new XMLHttpRequest();

        success = true;
      }
      catch (microsoft) {
        // Microsoft

        var httpIds = new Array('MSXML2.XMLHTTP.5.0',
                                'MSXML2.XMLHTTP.4.0',
                              'MSXML2.XMLHTTP.3.0',
                              'MSXML2.XMLHTTP',
                              'Microsoft.XMLHTTP');

  for ( var i = 0; i < httpIds.length  &&  !success; i++ ) {
    try {
      request = new ActiveXObject(httpIds[i]);

      if ( request != null )
        success = true;
    }
    catch (e) {
      alert("no IE request object!");
    }
  }
}

if ( request == null )
  alert("Error creating request object!");

// RSS formatter

function formatRSSdata(divname,response)
{
  var html = "";

  var docElt = response.documentElement;

  // if this does not work in IE, the content-type
  //  in the header was likely not set to "text/xml".

  var items = docElt.getElementsByTagName('item');

  for ( var i = 0; i < items.length; i++ ) {
    var title = items[i].getElementsByTagName('title')[0];

    var link = items[i].getElementsByTagName('link')[0];

    html += "<b> <a href='" + link.firstChild.data + "'>" +
             title.firstChild.data + "</a> </b> <br>";

    var cbDetails = document.getElementById("cbDetails");

    if ( cbDetails.checked ) {
      var desc = items[i].getElementsByTagName('description')[0];

      html += "<font size='-1'>" +
               desc.firstChild.data +
               "</font>";
    }
  }
```

```
    var targetDiv = document.getElementById(divname);
    targetDiv.innerHTML = html;
  }

  // dynamic Ajax RSS news feed reader.  Every Window object (see above comment
  //  in httpRequest()) has a document property, which represents the HTML
  //  document displayed in the window (p. 199 of "JavaScript" by Flanagan).

  function getRSSfeed()
  {
    // get selected RSS feed:

    var lblFeeds = document.getElementById("lblFeeds");

    if ( lblFeeds.value != null ) {
      httpRequest("GET",lblFeeds.value,true);
    }
  }

  // event handler for updating Web pages with response from Web server;
  //  uses DOM Document object of Web browser.

  function handleResponse()
  {
    if ( request.readyState == 4 ) {
      if ( request.status == 200 ) {
        // get the response from the server
        var customerAddress = request.responseText;

        // update the HTML Web <form>
        document.getElementById("address").value = customerAddress;
      }
    }
  }

  // initialize request object that has already been created

  function initRequest(requestType,requestURL,syncOrAsync)
  {
    try {
      // tell browser the function to run when request object
      //  ready state changes.
      request.onreadystatechange = handleResponse;

      // connect to Web server and communicate
      request.open(requestType,requestURL,syncOrAsync);

      // actual request to connect to Web server
      //  (server needs no data).
      request.send(null);
    }
    catch (err) {
      alert("The application cannot connect to server!");
    }
  }

  /* the request object communicates with the Web server.

     Parameters:
      requestType - GET or POST
      requestURL  - URL of server program; note that if no domain
                    name given, then request goes to same Web server
      syncOrAsync - synchronous (false) or asynchronous (true) request
   */

  function httpRequest(requestType,requestURL,syncOrAsync)
  {
    // initialize request object
    initRequest(requestType,requestURL,syncOrAsync);
  }

  //

  function createMarker(latlng,html)
  {
    html = '<div style="white-space:nowrap;">' + html + '</div>';

    var marker = new GMarker(latlng);

    GEvent.addListener(marker,"click",function()
                                     {
                                       GLog.write('enter marker click handler');
                                       GLog.writeHtml(html);
                                       marker.openInfoWindowHtml(html);
                                       GLog.write('exit marker click handler');
                                     });
    return marker;
  }

  //

  function initMap(container)
  {
    var zoomEvent;

    GLog.write('enter initMap()');

    if ( typeof(GMap2) != "undefined" ) {
      // Maps API version >= 2.36
      map = new GMap2(container);
```

```
          zoomEvent = 'zoomend'
        }
        else {
          // Maps API version <= 2.35
          map = new GMap2(container);

          zoomEvent = 'zoom'
        }

        map.addControl(new GLargeMapControl());
        map.addControl(new GMapTypeControl());
        map.setCenter(new GLatLng(33.2130,43.4620),13);
//      map.setCenter(new GLatLng(37.4419,-122.1419),13);

        GEvent.addListener(map,'moveend',
         function()
         {
           GLog.write('moveend: ' + map.getCenter().toUrlValue() + ' zoom: '+ map.getZoom(),'blue')
         });

        GEvent.addListener(map,zoomEvent,
         function(a,b)
         {
           GLog.write(zoomEvent + ': from ' + a + ' to ' + b,'blue')
         });

        GEvent.addListener(map,'maptypechanged',
         function()
         {
           GLog.write('maptypechanged: ' + map.getCurrentMapType().getName(),'blue')
         });

        var latlng = new GLatLng(33.2130,43.4620);
        var marker = createMarker(latlng,'33.2130 N lat and 43.4620 W long');
//      var latlng = new GLatLng(37.4419,-122.1419);
//      var marker = createMarker(latlng,'37.4419 N lat and 122.1419 E long');
//      var marker = createMarker(latlng,'Welcome to Version 2<br>of the Google Maps API');

        map.addOverlay(marker);
        GLog.write('exit initMap()');
      }

      //

      function initPage()
      {
        GLog.write('enter initPage()');
        GLog.writeUrl(window.location.href);
        GLog.writeHtml('This HTML contains <strong>strong</strong>, <b>bold</b>, <strike>strike</strike>, <i>italic</i>,

        if ( !GBrowserIsCompatible() ) {
            alert("Your browser may not be compatible with the Google Maps system.\nPlease visit http://maps.google.com for
          }
          else {
            var mapDiv = document.getElementById("mapDiv");

            initMap(mapDiv);
          }

          GLog.write('Here is the URL to G_DEFAULT_ICON image');
          GLog.writeUrl(G_DEFAULT_ICON.image);
          GLog.writeHtml("This is a test <b>HTML</b> formated message<br />in two lines.");
          GLog.write('exit initPage()');
        }

        //

        function appletValue()
        {
          document.myForm.q.value = document.myApplet.getHello();
          return true;
        }

        //

        function onLoad()
        {
          initPage();

          app = new _app();
        }

        //

        function _app()
        {
//          this.myMap = null;
//          this.markerList = new Array();

          // a map

//          if ( GBrowserIsCompatible() ) {
//            this.myMap = new GMap2(document.getElementById("mapDiv"));
//            this.myMap.setCenter(new GLatLng(37.3861,-122.083),14);
//          }

//          this.myMap.addControl(new GSmallMapControl());

          // create search control, options

          var searchControl = new GSearchControl();
        var options = new GsearcherOptions();

        options.setExpandMode(GSearchControl.EXPAND_MODE_OPEN);

        // full set of Google services (2/21/07)

        var webSearch   = new GwebSearch();
        var newsSearch  = new GnewsSearch();
        var localSearch = new GlocalSearch();
        var videoSearch = new GvideoSearch();
        var blogSearch  = new GblogSearch();
        var bookSearch  = new GbookSearch();

        // need var for each site in a Google Web search;
        //  US sites and foreign sites

        var webSearch1 = new GwebSearch();
        var webSearch2 = new GwebSearch();
        var webSearch3 = new GwebSearch();
        var webSearch4 = new GwebSearch();
        var webSearch5 = new GwebSearch();
        var webSearch6 = new GwebSearch();
        var webSearch7 = new GwebSearch();
        var webSearch8 = new GwebSearch();
        var webSearch9 = new GwebSearch();

        webSearch1.setUserDefinedLabel("Web(CNN)");
        webSearch1.setSiteRestriction("www.cnn.com");

        webSearch2.setUserDefinedLabel("Web(USA Today)");
        webSearch2.setSiteRestriction("www.usatoday.com");

        webSearch3.setUserDefinedLabel("Web(CBS)");
        webSearch3.setSiteRestriction("www.cbsnews.com");

        webSearch4.setUserDefinedLabel("Web(ABC)");
        webSearch4.setSiteRestriction("abcnews.go.com");

//        webSearch5.setSiteRestriction("www.cnn.com");
//        webSearch6.setSiteRestriction("www.cnn.com");

        webSearch7.setUserDefinedLabel("Web(BBC)");
        webSearch7.setSiteRestriction("news.bbc.co.uk/2/hi/middle_east");

        webSearch8.setUserDefinedLabel("Web(Reuters)");
        webSearch8.setSiteRestriction("today.reuters.com/news");

        webSearch9.setUserDefinedLabel("Web(KUNA)");
        webSearch9.setSiteRestriction("http://www.kuna.net.kw");

        // make use of Google Web and Google News searchers
```

```
          searchControl.addSearcher(webSearch1);
          searchControl.addSearcher(webSearch2);
          searchControl.addSearcher(webSearch3);
          searchControl.addSearcher(webSearch4);
//          searchControl.addSearcher(webSearch5);
//          searchControl.addSearcher(webSearch6);
          searchControl.addSearcher(webSearch7);
          searchControl.addSearcher(webSearch8);
          searchControl.addSearcher(webSearch9);

          searchControl.addSearcher(newsSearch,options);

          // for Google Maps gadget
          searchControl.addSearcher(localSearch);

          // add Google Video searcher to controller
//          searchControl.addSearcher(videoSearch);

          // add Google Blog searcher to controller
//          searchControl.addSearcher(blogSearch);

          // add Google Book searcher to controller
//          searchControl.addSearcher(bookSearch);

          // tell the searcher to draw itself and where to attach
          searchControl.draw(document.getElementById("searchControlDiv"));

          // search control callbacks

          searchControl.setSearchCompleteCallback(this,onSearchComplete);
          searchControl.setSearchStartingCallback(this,onSearchStarting);
          searchControl.setOnKeepCallback(this,onKeep);

          // execute an inital search
          searchControl.execute(searchString);
        }

        //

        function onSearchComplete(searchControl,searcher)
        {
          // if we have local search results, put them on the map

          if ( searcher.results  &&  searcher.results.length > 0 ) {
            alert(searcher.results.length);

            for ( var i = 0; i < searcher.results.length; i++ ) {
              var result = searcher.results[i];

              // Google News service
```

19

```
      if ( result.GsearchResultClass == GnewsSearch.RESULT_CLASS )
        alert("Google News URL = " + result.url);

      // Google Web service

      if ( result.GsearchResultClass == GwebSearch.RESULT_CLASS )
        alert("Google Web URL = " + result.url);

      // Google Local service

      if ( result.GsearchResultClass == GlocalSearch.RESULT_CLASS ) {
        alert("Google Local URL = " + result.url);

        var markerObject = new Object();

        markerObject.result = result;

        markerObject.latLng = new GLatLng(parseFloat(result.lat),parseFloat(result.lng));

        markerObject.gmarker = new GMarker(markerObject.latLng);

        var clickHandler = method_closure(this,onMarkerClick,[markerObject]);

        GEvent.bind(markerObject.gmarker,"click",this,clickHandler);

        this.markerList.push(markerObject);
        this.myMap.addOverlay(markerObject.gmarker);
        result.__markerObject__ = markerObject;
      }

      // Google Video service

      if ( result.GsearchResultClass == GvideoSearch.RESULT_CLASS )
        alert("Google Video URL = " + result.url);

    }
    this.onMarkerClick(this.markerList[0]);
  }
}

//

function onSearchStarting(searchControl,searcher,query)
{
  // close the info window and clear markers

  this.myMap.closeInfoWindow();

  for ( var i = 0; i < this.markerList.length; i++ ) {
    var markerObject = this.markerList[i];
```

```
          this.myMap.removeOverlay(markerObject.gmarker);
        }

        this.markerList = new Array();
      }

      //

      function onKeep(result)
      {
        if ( result.__markerObject__ ) {
          markerObject = result.__markerObject__;
          this.onMarkerClick(markerObject);
        }
      }

      //

      function onMarkerClick(markerObject)
      {
        this.myMap.closeInfoWindow();

        var htmlNode = markerObject.result.html.cloneNode(true);

        markerObject.gmarker.openInfoWindow(htmlNode);
      }

      //

      function method_closure(object,method,opt_argArray)
      {
        return function()
               {
                 return method.apply(object,opt_argArray);
               }
      }
     //]]>
    </script>
  </head>

  <body onload="onLoad()">
    <img src="rtna.bmp" width="850" hspace="20" vspace="40"/>

    <h1>
      using the Google AJAX Search API
    </h1>

    <table>
      <tr>
        <td class="searchControl">
```

```
          <div id="searchControlDiv">
            Loading...
          </div>
        </td>
      </tr>
      <tr>
        <td class="map">
          <div id="mapDiv">
            Loading...

            <!-- Google Map gadget -->

          </div>
        </td>
      </tr>

      <tr>
        <td>
          &nbsp;
        </td>
      </tr>

      <tr>
        <td>
          <a href="http://www.lat-long.com/">
            city, state(USA)
          </a>
        </td>

        <td>
          or
        </td>

        <td>
          <a href="http://www.batchgeocode.com/lookup/">
            street address, city, state(USA)
          </a>
        </td>
      </tr>
    </table>
  </body>
</html>
```

Appendix B. Content Extraction Software

The following Java code is executed for a particular URL. Ideally it should, and will, be a <script> in the RTNA XHTML document.

This appendix appears in its original form, without editorial change.

```
package mil.army.arl.rtna;

/*
 * author: John Richardson (jrichardson@arl.army.mil)
 *
 * This product contains software developed by Andy Clark.
 * http://people.apache.org/~andyc/neko/doc/html/
 * CyberNeko HTML Parser
 */

import org.cyberneko.html.parsers.DOMParser;
import org.w3c.dom.Node;

public class HTMLDOMParser {
  public static String elementFilter (Node node)
  {
    String article = new String();

    Node child = node.getFirstChild();

    String nodeName;
    Node tmp;

    while ( child != null ) {
      nodeName = child.getNodeName();
      tmp = child.getNextSibling();

      if ( nodeName.compareTo("#text") == 0 )
        article += (child.getTextContent());                    // text content
      else if ( nodeName.compareTo("#comment") == 0   ||        // comment tags
                nodeName.compareTo("SCRIPT") == 0     ||        // programming tags
                nodeName.compareTo("NOSCRIPT") == 0   ||
                nodeName.compareTo("OBJECT") == 0     ||
                nodeName.compareTo("PARAM") == 0      ||
                nodeName.compareTo("HEAD") == 0       ||
                nodeName.compareTo("TITLE") == 0      ||
                nodeName.compareTo("META") == 0       ||        // meta info tags
                nodeName.compareTo("BASE") == 0       ||
                nodeName.compareTo("IMG") == 0        ||        // image tags
                nodeName.compareTo("MAP") == 0        ||
                nodeName.compareTo("AREA") == 0       ||
                nodeName.compareTo("UL") == 0         ||        // lists tags
                nodeName.compareTo("OL") == 0         ||
                nodeName.compareTo("LI") == 0         ||
                nodeName.compareTo("DL") == 0         ||
                nodeName.compareTo("DT") == 0         ||
                nodeName.compareTo("DD") == 0         ||
                nodeName.compareTo("FORM") == 0       ||        // input tags
                nodeName.compareTo("INPUT") == 0      ||
                nodeName.compareTo("TEXTAREA") == 0   ||
                nodeName.compareTo("BUTTON") == 0     ||
                  nodeName.compareTo("SELECT") == 0     ||
                  nodeName.compareTo("OPTGROUP") == 0   ||
                  nodeName.compareTo("OPTION") == 0     ||
                  nodeName.compareTo("LABEL") == 0      ||
                  nodeName.compareTo("FIELDSET") == 0   ||
                  nodeName.compareTo("LEGEND") == 0 )
          node.removeChild(child);
        else
          article += elementFilter(child);

        child = tmp;
      }

      return article;
    }

    public static Node getHTMLDOMDocument (String target)
                        throws Exception
    {
      DOMParser parser = new DOMParser();

      parser.parse(target);

      Node doc = parser.getDocument();

      return doc;
    }

    public static void main(String args[])
                        throws Exception
    {
      // provide URL as argument
      System.out.println(HTMLDOMParser.elementFilter(HTMLDOMParser.getHTMLDOMDocument(args[0])));
    }
  }
```

INTENTIONALLY LEFT BLANK.

List of Symbols, Abbreviations, and Acronyms

Ajax	asynchronous JavaScript and XML
API	application programming interface
ARL	U.S. Army Research Laboratory
CSS	cascading style sheet
DOM	document object model
GUI	graphical user interface
HTML	hypertext markup language
RTNA	Real-Time News Analysis
SNA	Social Network Analysis
XHTML	extensible hypertext markup language
XML	extensible markup language

NO. OF COPIES	ORGANIZATION
1 (PDF ONLY)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FORT BELVOIR VA 22060-6218
1	US ARMY RSRCH DEV & ENGRG CMD SYSTEMS OF SYSTEMS INTEGRATION AMSRD SS T 6000 6TH ST STE 100 FORT BELVOIR VA 22060-5608
1	DIRECTOR US ARMY RESEARCH LAB IMNE ALC IMS 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CI OK TL 2800 POWDER MILL RD ADELPHI MD 20783-1197
	ABERDEEN PROVING GROUND
1	DIR USARL AMSRD ARL CI OK TP (BLDG 4600)

NO. OF COPIES	ORGANIZATION
1	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CI CB C VOSS 2800 POWDER MILL RD ADELPHI MD 20783-1197
	ABERDEEN PROVING GROUND
13	DIR USARL AMSRD ARL CI CT B BROOME M THOMAS A NEIDERER (6 CPS) J RICHARDSON (2 CPS) J FORESTER J OMAY A WETMORE

INTENTIONALLY LEFT BLANK.